IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application. No: 10/676,634	§	Examiner:	Augustine, Nicholas
Filed: October 1, 2003	§		
Inventor(s):	§	Group/Art Unit:	2179
Luis M. Gomes, Madras S. Mohanasundaram	§	Atty. Dkt. No:	5150-82801
	§		
Title: EDITABLE DATA TOOLTIPS	§		
	§		
	§		
	§		

REPLY BRIEF

Dear Sir/Madam:

Further to the Examiner's Answer of October 6, 2009, Appellant presents this Reply Brief. Appellant respectfully requests that this Reply Brief be considered by the Board of Patent Appeals and Interferences.

III. STATUS OF CLAIMS

Claims 1 and 6-23 are pending in the case. Claims 1 and 6-23 stand rejected under 35 U.S.C. § 103(a) and are the subject of this appeal.

REMARKS

Arguments

Regarding the Examiner's *new* arguments directed to the section 103 rejection of claims 1 and 6-23 over Deutscher et al (US Pat. Pub. 2004/0001106, "Deutscher"), in view of Hampapuram et al. (US Pat. Pub. 2004/0221262, "Hampapuram"), Appellant presents the following arguments:

Claims 1, 10, 11, 12, 17, 18, 19, 20, 21

Regarding the limitation: **displaying source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code, receiving first user input hovering a mouse cursor over an expression in the source code during execution of the program;** and **in response to said hovering the mouse cursor over the expression, automatically displaying a GUI element proximate to the expression, wherein the GUI element includes a value of the expression**, as recited in claim 1, the Examiner's Answer admits that "Deutscher does not specifically teach the exact term 'source code'", but then argues that Deutscher's script grid "performs the same functionality" as program source code. First, Appellant respectfully submits that Deutscher's script grid does *not* "perform the same functionality" as program source code, as one of skill in the programming arts readily understands–e.g., Deutscher's script grid is not compilable to generate an executable program, but rather, is a means/GUI for specifying input data and script names to be included in a presentation data file for a presentation engine (program), as Deutscher makes clear. Secondly, "performing the same functionality" is not a standard by which patentability is determined, as the Examiner is certainly aware–many, if not most, patents are directed to new and inventive ways to provide known functionalities, e.g., new ways to control machinery, new ways to produce products, etc.

For example, the Examiner argues that the "Script grid is read by the computer to perform the final outcome of the presentation, also described as a set of instructions read by a computer to perform a function.", and that "Each may be written by a user and composed/edited but essentially they are both provided to provide instructions to a computing processor to perform a function as detailed by the instructions as defined by a user/programmer". Appellant submits that the Examiner's analysis and characterization is incorrect and improper.

For example, Appellant notes that data files are "read by a computer", and certainly determine output, and may be composed/edited by a user, but clearly do not qualify as source code for a program. As Appellant has previously explained in detail, Deutscher is directed to a multimedia presentation production system in which presentation data are separated from presentation logic (see, e.g., [0008]). Per [0009], Deutscher discloses a tool consisting of "***graphic user interfaces* for easily entering the data associated with the rich media presentation**", where "**the tool is also designed to make it simple and easy to edit the data of the underlying schema**." Per [0017]-[0018], **among the various data entry graphical user interfaces (GUIs)** disclosed are "**data entry grids**", which are displayed in a presentation tool window, and "**allow the user to enter the information into the presentation data file and the master track media file that is needed to drive the timeline of the presentation**. **The first of these grids, which is displayed by default once the master track program has been imported, is the scripts grid**. The scripts grid is where the user enters events called script commands that will be triggered during the playback of the presentation." Per Deutscher, and as described in more detail in section 2.6.1 ([0137] – [0144]), the scripts grid is "**essentially a scheduling table** where the user enters events (sometimes referred to as script commands) that will be triggered during the playback of the presentation." The **entered data in the various data grids (including the scripts grid) are saved to the presentation *data* file**, which is then referenced by the presentation system to present the multimedia presentation. **Thus, a scheduling table is *not* source code that is compilable to generate an executable program, but rather is a means for specifying *input data***, including the names of scripts to be invoked by the engine.

Appellant further notes that a file that lists the names of scripts or even subprograms to be executed by a presentation engine would still not be considered to be *program source code*. As explained above, Deutscher is quite clear that the cited script grids are GUIs for entering data, not compilable program source code, nor, more specifically, source code that is compilable to generate the executable program. The Examiner has not explained where Deutscher describes compiling the cited GUIs (script grids) to generate the executing program–Deutscher's presentation engine, but rather, seems to simply ignore these recited limitations of claim 1, which is improper.

As also explained previously, one of skill in the programming arts understands that "source code" is a term of art in the programming domain and refers specifically to program instructions in a programming language that are compilable to produce executable code that may be run on a processor. This is emphasized in the independent claims by the limitation: "**wherein**

the executing program was compiled from the source code", which is certainly not the case with the cited script grid. Deutscher does not describe any of the data grids as source code for an executing program. Per cited paragraph 137, the program that is paused to allow for Deutscher's scripts grid editing is "a video or audio program that is designated as the master track", *not* an executable program for which the scripts grid is source code.

Appellant respectfully reminds the Examiner that *every word in the claim* is to be given consideration, and that the terms of the claim are to be interpreted in light of the Specification. In other words, terms of art should not be re-interpreted contrary to the Specification. A scheduling table (Deutscher's own description of the script grid) used by an executing program to manage multi-media presentations does not "perform the same function" as source code that is compiled to generate the executable program. Note that substituting either of these elements for the other in their respective inventions would render the respective inventions inoperable, and so they clearly do *not* perform the same function, and are not equivalent. One of ordinary skill in the programming arts would readily understand that an editing tool and a data file used by an executing program (the presentation viewer) are *not* source code for the executable program, **wherein the executing program was compiled from the source code**, as recited in claim 1. Per Deutscher, the script grid is a data entry grid that allows the user to enter information (i.e., *data*) into the presentation data file, and the presentation data file is (obviously) a *data file*, which is used by the presentation viewer (the executable program) to present a presentation. Neither the script grid, nor the data produced by the script grid, nor the presentation data file, is compiled into executable code, i.e., executable program instructions, contrary to the Examiner's assertions.

The Examiner's Answer again cites that paragraphs 137, 155, and 180, as well as Figures 13 and 17, arguing that "Deutscher provides that the script grid is partially made up of transcription's [sic] that can be added and are part of the transcription grid". As explained previously, these citations are directed to editing transcription entries via a text entry dialog box, where the user selects a data field, e.g., a time code field, and overwrites the data therein. Again, these citations in no way teach that the transcription entries or the transcription grid in general comprise source code that is compiled to generate the executable program instructions of the presentation viewer, nor that the transcription entries are edited during execution of the presentation viewer. Moreover, in paragraph [0152], Deutscher states that the default language for the transcriptions is *English*, which is decidedly *not* a programming language.

The Examiner's Answer continues to mischaracterize and misinterpret the nature of the "for" loop shown in Figure 17. As explained previously (and repeatedly), the program code shown is actually the contents of the transcription *data*; the transcription is human-readable text,

not compilable program source code. More specifically, **the presented FOR loop is text for a tutorial on programming**, *not* actual program source code. As Deutscher makes abundantly clear, the contents of the grids are data, not source code that is compilable to executable program instructions.

The Examiner's Answer then cites Hampapuram, arguing that this reference "teaches the use of user definable information being that of source code from an executable program", specifically citing paragraphs 7, 20, and 22. Appellant has already addressed these citations in the Appeal Brief, and the Examiner has not provided any new arguments. Appellant again notes that paragraph 7 is directed to static or edit/compile time expansion of macros, not at runtime, and thus is not germane to these features. Again, *nowhere does Hampapuram describe the macro expansions occurring at runtime.* Cited paragraph 20 simply states that the software development tool under which Hampapuram's static macro expansion is performed may be any of "a source code browser, debugger, static analysis tool, or integrated development environment, among others"; however, this has no bearing on when Hampapuram's macro expansion occurs. More importantly, cited paragraph 22 particularly explains that the macro expansion occurs at compile time, i.e., *not* during execution of the program.

Appellant respectfully requests that the Examiner give proper consideration to the functionality resulting from the particular arrangement of features and limitations claimed, which Appellant respectfully submits is not provided by the alleged combination of Deutscher and Hampapuram. For example, combining Deutscher's script grids (which are GUIs for entering data for a multimedia presentation engine) with Hampapuram's edit-time expansion of macros would not result in the features and limitations, nor functionality, expressed in claim 1. More specifically, Deutscher and Hampapuram, taken singly or in combination, fail to teach displaying the source code of a program during execution of the program.

The Examiner's Answer further argues that Deutscher's "program being used in the system is a debugger program", that "Deutscher describes how the user is able to interact and bring up a graphical user interface for editing user definable information", and that Deutscher's "system is open to have its control methods work for any program like a debugger having source code associated with it as taught by Hampapuram"; Appellant notes that Deutscher's executing program is a multimedia presentation engine, and that any editing/debugging in Deutscher is directed to development of *data files*, not an executable program. Moreover, Hampapuram's macro expansion is not performed at runtime. More generally, Appellant respectfully notes that any invocation of a GUI or display of an expression or macro disclosed in Deutscher and Hampapuram in response to mouse hovering occurs at edit time, i.e., statically, not at runtime.

More detailed analysis of the cited references regarding the "mouse hovering" related functionality is provided in the Appeal Brief.

Thus, the cited art, taken singly or in combination, fails to teach or suggest these claimed features.

Regarding the limitation: **receiving second user input to the GUI element modifying the displayed value, thereby specifying a new value for the expression; and setting the expression in the program to the new value in response to the second user input, wherein the program continues execution in accordance with the new value of the expression**, as recited in claim 1, the Examiner's Answer argues that Deutscher's editing of presentation data, then executing the presentation engine using the edited data somehow teaches Appellant's claimed dynamic (i.e., at runtime) editing of the value of a program expression, where the program continues to execute, but using the new value citing paragraphs 137, 155, and 200-201 and Figures 13, 17, and 26B.

Appellant has studied these citations carefully and respectfully disagrees.

As explained previously, per paragraph 137, the program that is paused to allow for Deutscher's scripts grid editing is "a video or audio program that is designated as the master track", *not* an executable program; i.e., Deutscher's editing is of data, and is static. The Examiner's Answer emphasizes this point, saying that "Upon the user completing the edited changes [of the presentation data] the user is able to execute the current document that was edit [sic] to see the changes"; note that 1) documents are not generally considered to be executable programs; 2) Deutscher's executable program is the presentation engine, and is never described as being edited; 3) the presentation engine / program is not executing when the data are being edited.

Paragraph 155 and Figure 17 are directed to editing transcription entries, i.e., data, prior to execution of the presentation engine, and, as explained above and in the Appeal Brief, neither teaches nor suggests modifying the value of an expression in the source code of a program *while the program* (which was compiled from the source code) *is executing*, where the program *continues to execute, but using the modified value*, as claimed.

As also explained in the Appeal Brief, Figure 13 shows a text entry dialog box whereby the user can overwrite textual data for the script grid data table, but, as noted previously, the data do not include source code that is compilable to executable program instructions of the presentation viewer.

Again, these citations in no way teach that the transcription entries or the transcription grid in general comprise source code that is compiled to generate the executable program

instructions of the presentation viewer, nor that the transcription entries are edited during execution of the presentation viewer.

Thus, Deutscher and Hampapuram fail to disclose displaying a value of an expression *in source code of an executing program*, i.e., at runtime, nor modifying the value of such an expression in the source code of an executing program, *where the program continues execution in accordance with the new value of the expression*.

Similarly, Hampapuram's macro expansions do not teach or suggest modifying the value of an expression in the source code of an executing program, where the program continues execution in accordance with the new value of the expression, and are not performed during execution of the program.

Claims 6, 7, 8

Regarding the limitation **wherein the GUI element is context sensitive**, as recited in claim 6 (and 7 and 8), Applicant respectfully notes that, as explained previously, the cited GUIs of paragraph 137 and Figure 13 are not context sensitive, but rather are specific GUIs for the grids being displayed and edited (e.g., scripts grid, and markers grid, respectively), e.g., "where only numbers are accepted (item 1304)", and are nowhere describes as changing or responding dynamically based on the nature of an expression indicated by the user (more specifically, by hovering). Thus, these figures do not, and cannot, disclose context sensitive GUIs, nor, more particularly, with the features of claims that are dependent from claim 6.

Claim 9

Regarding the limitation: **wherein the specified format is specified via a second GUI element in the GUI**, as recited in claim 9, Appellant respectfully notes that the cited GUI elements of Deutscher are type-specific fields with identifying labels, e.g., specific edit fields for editing marker attributes in a markers grid, specifically, fields for time code values and transcript text. However, none of these GUI elements is described as a context sensitive GUI whereby the *data format for another field in the GUI may be specified*.

Thus, the cited art fails to teach or suggest this feature.

Claims 13 – 16 (which Applicant notes are argued separately)

The Examiner's Answer argues "that Deutscher was editing information to be presented while being able to preview the content (dynamic editing)." This is both incorrect and irrelevant. First, Deutscher does not disclose editing values of expressions in source code of an executing program *during execution of the program*, as discussed at length above and in the Appeal Brief. Second, Deutscher's editing of presentation data is nowhere described as occurring during execution of the presentation engine; rather, after *editing the data*, the presentation engine is run to view the results.

Thus, the cited art fails to teach these features.

Claims 22 and 23

Regarding these two claims (which Appellant has argued separately), in response to Appellant's assertion "that Deutscher does not teach evaluating the expression to determine the value of the expression", the Examiner's Answer argues that Deutscher teaches that "the interface element used to edit information depicted in figure 13 at least, is shown as only being just large enough to display edit boxes and indication text."

Appellant respectfully submits that this is not germane to these claimed features nor Appellant's quoted assertion, at least because neither reference teaches evaluating an expression in source code of an executing program to determine the value of the expression (at runtime). As explained above, Deutscher teaches editing *data* for use by a presentation engine (where, Appellant notes, the editing is not performed during execution of the program), and Hampapuram teaches static expansion (e.g., at compile time) of macros in a program, which, again, is not at runtime. Moreover, as explained previously, in Hampapuram, the tooltip displays the macro, the value of the expanded macro, *and* a sentence that contains both, e.g., "macro 'NULL' expands to:0", which clearly contains more than simply the value of the indicated macro (which is not an expression from the source code of an executing program anyway). Nor are Hampapuram's tooltips editable.

Thus, the cited art also fails to disclose these features.

For the foregoing reasons, it is respectfully submitted that the Examiner's rejection of claims 1 and 6-23 was erroneous, and reversal of the decision is respectfully requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above-referenced application(s) from becoming abandoned, Appellant(s) hereby petition for such extensions. The Commissioner is hereby authorized to charge any fees which may be required or credit any overpayment to Meyertons, Hood, Kivlin, Kowert & Goetzel P.C., Deposit Account No. 50-1505/5150-82801/JCH.

Respectfully submitted,

/Jeffrey C. Hood/
Jeffrey C. Hood, Reg. #35198
ATTORNEY FOR APPLICANT(S)

Meyertons Hood Kivlin Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800

Date: 2009-11-02 JCH/MSW